

# Computation of the signed distance function to a discrete contour on adapted triangulation.

C. Dapogny<sup>a,b</sup>, P. Frey<sup>c</sup>

<sup>a</sup>*Centre de Mathématiques Appliquées (UMR 7641), Ecole Polytechnique 91128 Palaiseau, France.*

<sup>b</sup>*Renault DREAM-DTAA Guyancourt, France.*

<sup>c</sup>*UPMC Univ Paris 06, UMR 7598, Laboratoire J.-L. Lions, F-75005 Paris, France.*

---

## Abstract

In this paper, we propose a numerical method for computing the signed distance function to a discrete domain, on an arbitrary triangular background mesh. It mainly relies on the use of some theoretical properties of the unsteady Eikonal equation. Then we present a way of adapting the mesh on which computations are held to enhance the accuracy for both the approximation of the signed distance function and the approximation of the initial discrete contour by the induced piecewise affine reconstruction, which is crucial when using this signed distance function in a context of level set methods. Several examples are presented to assess our analysis, in two or three dimensions.

*Keywords:* Signed distance function, Eikonal equation, level set method, anisotropic mesh adaptation,  $\mathbb{P}^1$ -finite elements interpolation.

---

## 1. Introduction

The knowledge of the signed distance function to a domain  $\Omega \subset \mathbb{R}^d$  ( $d = 2, 3$  in our applications) has proved a very valuable information in various fields such as collision detection [18], shape reconstruction from an unorganized cloud of points [41],[11], and of course when it comes to level set methods, introduced by Sethian and Osher [31] (see also [35] or [30] for multiple topics around the level set method), where ensuring the property of unitary gradient of the level set function is especially relevant.

We intend here to devise a numerical algorithm that allows the computation of the signed distance function to a *polyhedral* domain  $\Omega$ , with minimal assumptions or requirements. We only rely on the knowledge of a triangular mesh  $\mathcal{S}$  of its boundary  $\partial\Omega$ , which we suppose orientable (if not, the interior  $\Omega$  is ill-defined on the sole basis of its boundary). Making this assumption, we imply here that we do not rely on any knowledge of the exterior normal to the surface, nor do we suppose the mesh  $\mathcal{S}$  to be conformal. We do not want to make any

---

\*corresponding author: P. Frey

\*\*This author is partially supported by the Chair "Mathematical modelling and numerical simulation, F-EADS - Ecole Polytechnique - INRIA".

*Email addresses:* `dapogny@ann.jussieu.fr` (C. Dapogny), `frey@ann.jussieu.fr` (P. Frey)

kind of assumption on the triangular background mesh  $\mathcal{T}$  on which the computations are performed either, even if, of course, the quality of the solution may depend on it, as will be seen.

Regarding the computation of the signed distance function, there exist mainly two types of approaches, both of them being based on an approximation of the solution of the Eikonal equation (1). The first one consists in treating this equation as a stationary boundary value problem, starting from the knowledge of the distance in the elements of the computational grid or mesh which are close to the interface  $\partial\Omega$ , and propagating the information throughout the whole domain. The most notorious methods - such as the Fast Marching Method [34], [33], [25] or the Fast Sweeping Method [40], [29] - belong to this category, and rely on a local solver for the Eikonal equation, and on a marching method, meant to enforce the natural causality embedded in the equation. Another way of addressing the problem is to consider it as an unsteady problem and then to devise a propagation method for extending the signed distance field from the boundary of  $\Omega$  - see [37] or [38]. This is the point of view which is at stake in this paper. This leads to an efficient, easy to implement, and easy to parallelize method for computing the signed distance function to  $\Omega$ . The proposed method is presented in dimensions 2 and 3, but naturally extends to the general case.

What is more, when we want to use the signed distance function as an implicit function defining the initial domain  $\Omega$  from which we intend to evolve, the discrete interface obtained as the 0 level set of the discrete signed distance function can turn out to be quite different from the true interface  $\partial\Omega$ . Following the work presented in [14], we intend here to present an adaptation scheme, based on the signed distance function, to produce a background mesh adapted to the boundary  $\partial\Omega$  so as to improve in the meantime, the computation of the signed distance function to  $\Omega$  - at least in the areas of the computational domain where it is relevant - and the discrete numerical reconstruction of  $\Omega$ .

The remainder of this paper is organized as follows. In section 2, we review some general properties of the signed distance function, and among other things recall how it can be seen as the stationary state of the solution of the unsteady Eikonal equation, which leads us to studying the dynamics of this equation in section 3. From this study, we infer a numerical scheme for approximating the signed distance function in section 4. We then show in section 5.1 how the background triangular mesh on which the computation is held can be adapted so that both the approximation of the signed distance function and the discrete isosurface resulting from the process can be controlled and improved. We briefly discuss two interesting extensions of this work, to the problem of reinitialization of a level set function in section 6, and to the computation of the signed distance function to a domain in a Riemannian space in section 7. Numerical examples are eventually provided in section 8 to emphasize the main features of our approach.

## 2. Some preliminaries about the signed distance function

So as to get a better intuition about the signed distance function and the various difficulties it raises, let us briefly recall some 'classical' results around the topic.

**Definition 2.1.** Let  $\Omega \subset \mathbb{R}^d$  a bounded domain. The signed distance function to  $\Omega$  is the function  $\mathbb{R}^d \ni x \mapsto u_\Omega(x)$  defined by :

$$u_\Omega(x) = \begin{cases} -d(x, \partial\Omega) & \text{if } x \in \Omega \\ 0 & \text{if } x \in \partial\Omega \\ d(x, \partial\Omega) & \text{if } x \in \overline{c\Omega} \end{cases} ,$$

where  $d(\cdot, \partial\Omega)$  denotes the usual Euclidean distance function to the set  $\partial\Omega$ .

When studying the distance to such a bounded domain  $\Omega$ , the fundamental notion of *skeleton* plays a major role :

**Definition 2.2.** The skeleton of  $\partial\Omega$  is the set of points  $x \in \mathbb{R}^d$  such that the minimum in

$$d(x, \partial\Omega) = \min_{y \in \partial\Omega} \|x - y\|$$

is achieved for at least two distinct points of  $\partial\Omega$ .

Being 1-lipschitz, and owing to Rademacher's theorem (see for instance [15]), the function  $u_\Omega$  is almost everywhere differentiable. Actually, the following interesting proposition delivers a geometric characterization of the regularity of  $u_\Omega$ . Its proof can be found in [3].

**Theorem 2.1.** Let  $\Omega \subset \mathbb{R}^d$  be a bounded open set with smooth boundary ; then  $u_\Omega$  is smooth in a tubular neighbourhood  $U$  of  $\partial\Omega$ . Moreover, for any point  $x \in \mathbb{R}^d$ ,

- either  $x \in \partial\Omega$ , and then  $u_\Omega$  is differentiable at  $x$ , with

$$\nabla u_\Omega(x) = n(x), \text{ the unit normal vector to } \partial\Omega \text{ at } x$$

- or  $x \in \mathbb{R}^d \setminus \partial\Omega$ , then  $u_\Omega$  is differentiable at  $x$  if and only if  $x$  belongs to the complementary of the skeleton of  $\partial\Omega$ . In such a case, there exists a unique point  $p_x \in \partial\Omega$  such that  $d(x, \partial\Omega) = \|p_x - x\|$ , and the gradient of  $u_\Omega$  at  $x$  reads

$$\nabla u_\Omega(x) = \frac{p_x - x}{u_\Omega(x)}.$$

In particular,  $u_\Omega$  satisfies the so-called Eikonal equation at every point  $x$  where it is differentiable :

$$\|\nabla u_\Omega(x)\| = 1. \tag{1}$$

Unfortunately, theory happens to be scarce as for functions being solutions of a PDE almost everywhere. For this reason - and many others - it is much more convenient from a theoretical point of view to see  $u_\Omega$  as a *viscosity solution* of the Eikonal equation (see [15] again).

**Proposition 2.2.**  $u_\Omega$  is a solution of the Eikonal equation (1) in the sense of viscosity.

For the sake of completeness, let us mention that several approaches exist when it comes to studying the very degenerated equation (1) ; further developments lead to taking into account the boundary condition

$$u_\Omega(x) = 0 \text{ on } \partial\Omega$$

in the sense of viscosity (see [12]), but even so, equation (1) turns out to get 'too much solutions'.

Another way of thinking of  $u_\Omega$  consists of seeing it as the result of a propagation by means of an evolution equation : suppose  $\Omega$  is implicitly known as

$$\Omega = \{x \in \mathbb{R}^d; u_0(x) < 0\} \text{ and } \partial\Omega = \{x \in \mathbb{R}^d; u_0(x) = 0\}, \quad (2)$$

where  $u_0$  is a continuous function. Note that, in the theoretical framework, such a function  $u_0$  exists and is quite easy to construct by means of partitions of unity techniques. Then the function  $u_\Omega$  can be considered as the steady state of the so-called *redistanciation equation* or *unsteady Eikonal equation*

$$\begin{cases} \frac{\partial u}{\partial t} + \text{sgn}(u_0)(\|\nabla u\| - 1) = 0 & \forall t > 0, x \in \mathbb{R}^d \\ u(t = 0, x) = u_0(x) & \forall x \in \mathbb{R}^d \end{cases} \quad (3)$$

Formally speaking, this equation starts with an arbitrary (continuous) function  $u_0$  implicitly defining the domain  $\Omega$  and straightens it to the "best" function which suits that purpose -  $u_\Omega$ . For this reason, this equation was first introduced in [39] for redistancing a very stretched level set function arising in a calculus. The following important theorem (a proof can be found in [7] and [23]) makes these statements more precise :

**Theorem 2.3.** Let  $\Omega$  a bounded domain of  $\mathbb{R}^d$  implicitly defined by a continuous function  $u_0$  such that (2) holds. Define function  $u$ ,  $\forall x \in \mathbb{R}^d$ ,  $\forall t \in \mathbb{R}_+$ ,

$$u(t, x) = \begin{cases} \text{sgn}(u_0(x)) \inf_{\|y\| \leq t} (\text{sgn}(u_0(x))u_0(x + y) + t) & \text{if } t \leq d(x, \partial\Omega) \\ \text{sgn}(u_0(x))d(x, \partial\Omega) & \text{if } t > d(x, \partial\Omega) \end{cases} \quad (4)$$

Let  $T \in \mathbb{R}_+$ . Then  $u$  is the unique uniformly continuous viscosity solution of (3) such that, for all  $0 \leq t \leq T$ ,  $u(t, x) = 0$  on  $\partial\Omega$ .

Note that the exact formula (4) expresses the idea of a propagation of information from the boundary with constant unit speed. This feature will later be exploited to devise our numerical algorithm for computing the signed distance function to  $\Omega$ .



### 3. A short study of some properties of the solution to the unsteady Eikonal equation

So as to build a resolution scheme for the unsteady Eikonal equation (3), let us take a closer look to its dynamic in view of the previous section.

The main idea is to start with a function  $u_0$  which implicitly defines our domain  $\Omega$  in the sense that (2) hold and that we suppose continuous over  $\mathbb{R}^d$ , then to regularize it thanks to equation (3), considering the exact solution  $u$ , provided by formula (4), for the resulting Cauchy problem. For the sequel, it will prove convenient to assume moreover that the initial function  $u_0$  is an *overestimation* of the signed distance function  $u_\Omega$  to  $\Omega$ , except on a tubular neighbourhood  $U$  of  $\partial\Omega$ , where it is exactly  $u_\Omega$ , that is :

$$\forall x \in U, u_0(x) = u_\Omega(x) ; \forall x \in \mathbb{R}^d \setminus U, |u_0(x)| \geq |u_\Omega(x)| \quad (5)$$

We then have the following small result :

**Lemma 3.1.** *Assume  $\Omega$  is a bounded open set, with smooth boundary  $\partial\Omega$ . For any small enough time step  $dt > 0$ , denote  $t^n = ndt$ ,  $n \in \mathbb{N}$ . Suppose the initial function  $u_0$  satisfies (5) and denote  $u$  the solution of equation (3) provided by theorem 2.3. Then*

$$\begin{aligned} \forall x \in \mathbb{R}^d \setminus \Omega \quad u(t^{n+1}, x) &= \inf_{\|z\| \leq dt} (u(t^n, x+z) + dt) \\ \forall x \in \Omega \quad u(t^{n+1}, x) &= \sup_{\|z\| \leq dt} (u(t^n, x+z) - dt) \end{aligned} \quad (6)$$

*Proof.* First, thanks to the exact formula (4), it is clear that assumption (5) on  $u_0$  implies that for all  $t > 0$  and  $x \in \mathbb{R}^d$

$$|u(t, x)| \geq |u_\Omega(x)| \quad (7)$$

By symmetry, we restrict ourselves to showing the lemma in the case  $x \in \mathbb{R}^d \setminus \Omega$ .

Assume first that  $t^{n+1} < d(x, \partial\Omega)$ . Then,

$$\begin{aligned} u(t^{n+1}, x) &= \inf_{\|y\| \leq t^{n+1}} (u_0(x+y) + t^{n+1}) \\ &= \inf_{\|y\| \leq t^n, \|z\| \leq dt} (u_0(x+y+z) + t^n + dt) \\ &= \inf_{\|z\| \leq dt} (u(t^n, x+z) + dt). \end{aligned}$$

the last equality holding because  $d(x, \partial\Omega) > t^{n+1}$  implies that for all  $\|z\| \leq dt$ ,  $d(x+z, \partial\Omega) > t^n$ .

Suppose now that  $t^{n+1} \geq d(x, \partial\Omega)$ , and suppose  $dt$  is smaller than the width of the tubular neighbourhood  $U$  of  $\partial\Omega$  (on which  $u_\Omega$  is assumed to be smooth, according to theorem 2.1). We then have :

$$\begin{aligned} \inf_{\|z\| \leq dt} (u(t^n, x+z) + dt) &\geq \inf_{\|z\| \leq dt} (u_\Omega(x+z) + dt) \\ &= u_\Omega(x) \\ &= u(t^{n+1}, x). \end{aligned}$$

Conversely, either  $\|x\| \leq dt$ , then there exists  $\|z_0\| \leq dt$  such that  $(x + z_0) \in U$  and

$$\inf_{\|z\| \leq dt} (u(t^n, x + z) + dt) = u(t^n, x + z_0) = u_\Omega(x + z_0)$$

and the result follows thanks to the above arguments, or we can choose  $\|z_0\| \leq dt$  so that

$$\inf_{\|z\| \leq dt} (d(x + z, \partial\Omega)) = d(x + z_0, \partial\Omega) = d(x, \partial\Omega) - dt$$

and given that  $d(x + z_0, \partial\Omega) \leq t^n$ , we get eventually

$$\begin{aligned} \inf_{\|z\| \leq dt} (u(t^n, x + z) + dt) &\leq u(t^n, x + z_0) + dt \\ &= d(x + z_0, \partial\Omega) + dt \\ &= u_\Omega(x) \\ &= u(t^{n+1}, x) \end{aligned}$$

which concludes the proof.  $\square$

Suppose our computation is restricted to a bounded domain  $\mathcal{D}$  enclosing  $\Omega$ . We aim at computing function  $u(t^N, \cdot)$  over  $\mathcal{D}$  for  $N$  big enough so that  $t^N \geq \sup_{x \in \mathcal{D}} |u_\Omega(x)|$ . To this end, we use the iterative formulae provided by the previous lemma, transforming them a bit so as to get a decreasing sequence of functions, using the overestimation inequality (7) : introduce, for  $n = 0, 1, \dots$ , the sequence  $\widetilde{u}^n$  of continuous functions over  $\mathcal{D}$ , iteratively defined by  $\widetilde{u}^0 = u_0$ , and

$$\forall x \in \mathcal{D} \setminus \Omega, \widetilde{u}^{n+1}(x) = \inf (u(t^{n+1}, x), \widetilde{u}^n(x)) \quad (8)$$

$$\forall x \in \Omega, \widetilde{u}^{n+1}(x) = \sup (u(t^{n+1}, x), \widetilde{u}^n(x)) \quad (9)$$

From the above arguments, it follows that  $\widetilde{u}^n$  is a sequence of continuous functions over  $\mathcal{D}$  which converges pointwise to  $u_\Omega$ . Furthermore, from its definition, it is clear that for  $x \in \mathcal{D} \setminus \Omega$ ,  $\widetilde{u}^n(x)$  *decreases* from  $u_0(x)$  to  $u_\Omega(x)$  and that for  $x \in \Omega$ ,  $\widetilde{u}^n(x)$  *increases* from  $u_0(x)$  to  $u_\Omega(x)$ . This sequence of functions - computed thanks to the iterative process expressed in lemma 3.1 - is the one we will try and approximate in the next section so as to end up with the desired signed distance function.

#### 4. A numerical scheme for the signed distance function approximation

In this section, we propose a method for computing the signed distance function. The algorithm consists of two steps ; the first one is purely geometric and amounts to identifying the simplices of the background mesh which intersect contour  $\partial\Omega$ , and the second one is purely analytical and is based on an explicit numerical scheme for solving the unsteady Eikonal equation (3).

#### 4.1. Extending the signed distance function from the boundary

Given a polyhedral domain  $\Omega$ , known by means of a simplicial mesh  $\mathcal{S}$  of its boundary  $\partial\Omega$ , we intend to compute the signed distance function to  $\Omega$  at every node  $x$  of a simplicial mesh  $\mathcal{T}$  of a bounding box  $\mathcal{D}$ . More accurately, call  $\mathcal{K} := \{K \in \mathcal{T}; K \cap \partial\Omega \neq \emptyset\}$  the set of the simplices of  $\mathcal{T}$  which intersect the boundary  $\partial\Omega$ . We suppose that the signed distance function is initialized to its exact value in the nodes of those simplices  $K \in \mathcal{K}$ , and to a great value in the other nodes so that the initialization of the algorithm satisfies (5), at least in a discrete way ; see section 4.2 for the implementation of this initialization. We then expect to devise an iterative numerical scheme on basis of formulae (6) to extend this signed distance function to the whole bounding domain.

To achieve this, we propose to approximate  $u_\Omega$  by means of a finite element  $\mathbb{P}^1$  function on mesh  $\mathcal{T}$ . Let  $dt$  be a time-step,  $t^n = ndt$  ( $n = 0, \dots$ ) and  $u^n$  be a  $\mathbb{P}^1$  function intended as an approximation of the unique viscosity solution  $u$  of equation (3) at time  $t^n$ . We then iteratively compute  $u^n$  thanks to algorithm (1).

---

#### Algorithm 1 Extending the signed distance function field

---

1: Initialize the signed distance function  $u^0$  with :

$$\begin{cases} u^0(x) &= \text{exact signed distance function to } \Omega \text{ if } x \text{ belongs to a simplex of } \mathcal{K} \\ u^0(x) &= u_{MAX} \text{ otherwise} \end{cases}$$

2: **for**  $n = 1, \dots$  until convergence **do**

3:    $u^n(x) = u^{n-1}(x)$  for each node  $x$  of  $\mathcal{T}$

4:   **for** each simplex  $T$  of  $\mathcal{T}$  **do**

5:     **for** each node  $x$  of  $T$  which does not belong to a simplex in  $\mathcal{K}$  **do**

6:       **if**  $x \notin \Omega$  **then**

7:

$$u^n(x) = \min \left( u^{n-1}(x), u^{n-1}(x - \frac{\nabla(u^{n-1}|_T)}{\|\nabla(u^{n-1}|_T)\|} dt) \right) + dt \quad (10)$$

8:       **else**

9:

$$u^n(x) = \max \left( u^{n-1}(x), u^{n-1}(x + \frac{\nabla(u^{n-1}|_T)}{\|\nabla(u^{n-1}|_T)\|} dt) \right) - dt \quad (11)$$

10:     **end if**

11:   **end for**

12: **end for**

13: **end for**

14: **return**  $u^n$

---

This algorithm needs some clarifying comments : for each step  $t^n \rightarrow t^{n+1}$ , we intend to mimic formulae (6), except that the infimum and supremum appearing there are difficult to

compute in a discrete way. Assuming time step  $dt$  to be small enough, given for example a node of  $\mathcal{T}$ ,  $x \in \mathcal{D} \setminus \Omega$  and  $n \geq 0$  the simplest approximation of  $\inf_{\|z\| \leq dt} (u^n(x+z) + dt)$  is achieved considering the gradient of  $u^n$  at  $x$ . However, this gradient is a constant-per-simplex vector, and is obviously discontinuous at the interface between two adjacent simplices of  $\mathcal{T}$ . In particular, it is irrelevant to talk about its value at a node  $x$  of  $\mathcal{T}$ . The most natural way to discretize formula (6) is then

$$\inf_{\|z\| \leq dt} (u^n(x+z) + dt) \approx \inf_{T \in \text{Ball}(x)} u^n \left( x - \frac{\nabla(u^{n-1}|_T)}{\|\nabla(u^{n-1}|_T)\|} dt \right) + dt \quad (12)$$

where  $\text{Ball}(x)$  is the set of simplices of mesh  $\mathcal{T}$  containing  $x$  as vertex.

The rest of algorithm (1) is merely a discrete version of formulae (8) and (9).

**Remark** Actually, a formal study of the characteristic curves of equations (1) and (3) would have brought more or less the same numerical scheme. In that scope, the points  $x \in \mathcal{D}$  where  $u_\Omega$  fails to be smooth can be interpreted as the crossing points of different characteristic curves of Eikonal equation (1) ; see figure (1). At nodes  $x$  of mesh  $\mathcal{T}$  close to such kinks of the signed distance function, the discretization (12) expresses the idea that each one of these crossing characteristic curve is backtracked.

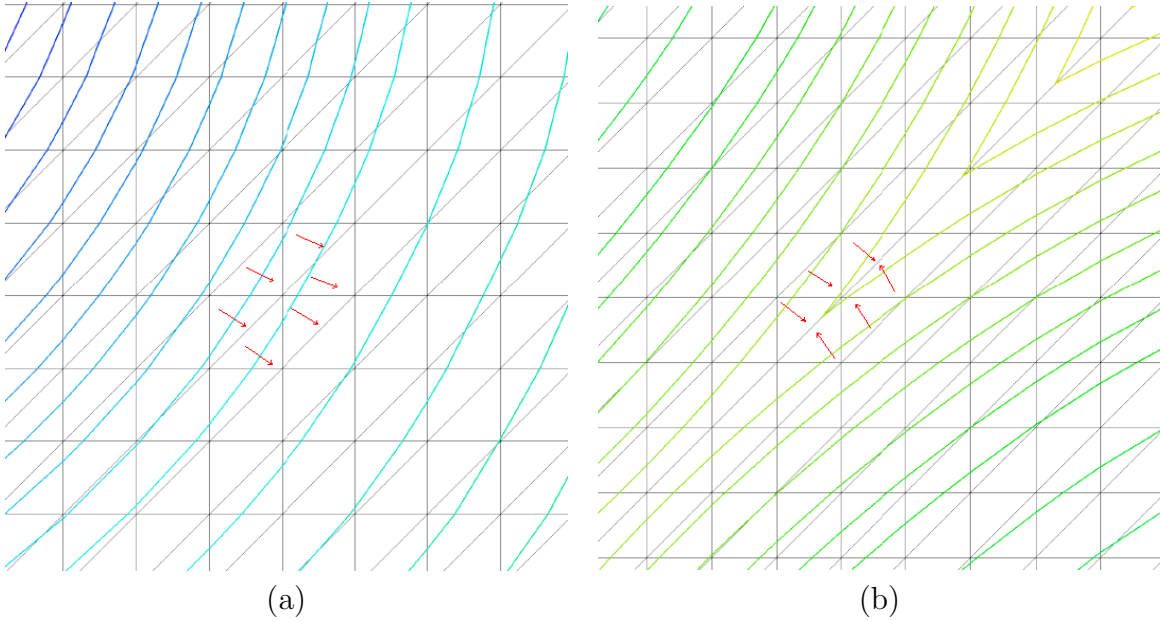


Figure 1: (a) Continuity of the gradient in the areas where the signed distance function is regular ; (b) discontinuity of the gradient at a node close to the skeleton.

#### 4.2. Initialization of the signed distance function near $\partial\Omega$

Before extending the signed distance function field from the boundary  $\partial\Omega$ , we need first to detect those simplices  $K \in \mathcal{T}$  which intersect  $\partial\Omega$ . This is achieved by scanning each surface triangle  $T \in \mathcal{S}$  in three dimensions (segment in two dimensions), storing a background mesh simplex  $K \in \mathcal{T}$  containing one of the three nodes of  $T$ , then travelling the background mesh from  $K$  by adjacency, advancing only through faces which intersect  $T$ . Figure 2 illustrates this step. A computationally efficient algorithm for the three-dimensional triangle-triangle overlap test, relying only on predicates, developed in [20] is used to this end.

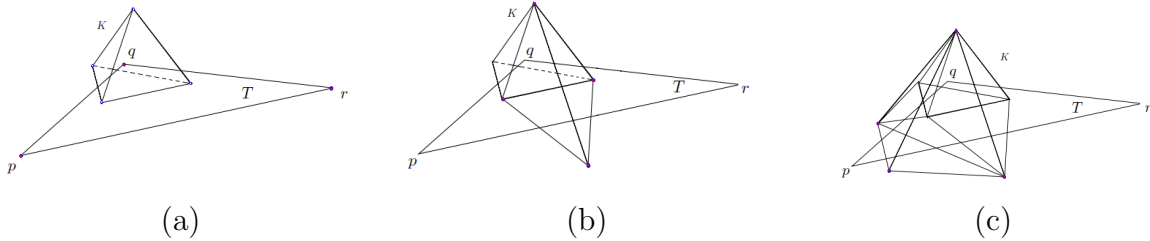


Figure 2: Recovery of the background simplices intersecting a triangle  $T = (pqr) \in \mathcal{S}$  : (a) one starts with a simplex  $K$  of  $\mathcal{T}$  containing one of the points  $p$ ,  $q$  or  $r$  then (b) marches through the faces of  $K$  which intersect  $T$  and (c) goes on, stopping when there is no more simplex of  $K$  to add.

Then, at each node of  $K$  which belongs to such an intersecting simplex, we initialize the (still unsigned) distance function to its exact value. See [24] for an efficient algorithm computing a point-triangle distance in three dimensions. In all the other nodes of  $\mathcal{T}$ , we assign an arbitrary large value (e.g. larger than the domain size).

This leaves us with initializing the sign. Surprisingly enough, this stage happens to be the most tedious one of the initialization process, all the more so as it is barely considered in the literature (see nevertheless [37] for another approach, based on an octree grid refinement). We propose here a purely logical algorithm (2) based on a progression by "layers", which relies on two piles *Layer* and *Boundary*, and an integer *Sign*. It is very similar to the classical coloring techniques used to recover the connected components of a configuration in a Delaunay meshing context [16].

So as to enhance the numerical efficiency of the proposed method, several improvements have been proposed to this algorithm. Note that the propagating scheme (1) is inherently parallel : at a given step, the operations carried out in a background simplex  $K$  are independant from those carried out in another such simplex. Furthermore, the time step  $dt$  must be chosen small enough at the beginning of the process, so that going back along the characteristic lines does not lead to crossing the interface  $\partial\Omega$  and picking irrelevant values. But after a certain amount of iterations, we can obviously increase this time step. Eventually, note that theorem (2.3) expresses the fact that the information propagates from  $\partial\Omega$  to the whole space with a unitary velocity - that is to say that, on the theoretical framework, at a given point  $x \in \mathbb{R}^d$ ,  $u(s, x) = u_\Omega(x, \partial\Omega)$  for all time  $s > d(x, \partial\Omega)$ . According to this observation,

---

**Algorithm 2** Signing the unsigned initial distance field

---

- 1: Initialize pile *Layer* with a simplex  $K \in \mathcal{T}$  which is in the "most external" connected component of the configuration (for instance a simplex in the corner of the bounding box).  
Initialize *Sign* to +1.
  - 2: **while** an element  $K$  in pile *Layer* as not been inspected **do**
  - 3:   Consider every neighbour  $K'$  of  $K$  : if  $K'$  has not yet been inspected, and does not belong to  $\mathcal{K}$ , put  $K'$  in *Layer* ; if  $K'$  belongs to  $\mathcal{K}$ , put it into pile *Boundary*.
  - 4: **end while**
  - 5: **while** an element  $K$  in *Boundary* has not yet been inspected **do**
  - 6:   Consider every neighbour  $K'$  of  $K$  : if  $K'$  has not yet been inspected, and belongs to  $\mathcal{K}$ , put  $K'$  in pile *Boundary* (this step ensures pile *Boundary* contains *all* the boundary elements corresponding to the ongoing layer).
  - 7: **end while**
  - 8: Put the current sign of *Sgn* in each vertex of each element of pile *Stratum*.
  - 9:  $Sign \leftarrow -Sign$ .
  - 10: Clean pile *Layer*.
  - 11: **for**  $K$  is in pile *Boundary* **do**
  - 12:   Consider every neighbour  $K'$  of  $K$  : if  $K'$  has not yet been inspected, put it into pile *Layer* : then we get all the new starting points of the new layer (possibly associated to different connected components of  $\Omega$ ).
  - 13: **end for**
  - 14: Clean pile *Boundary*.
  - 15: Go back to step 2.
- 

we decided to fix the values computed at a node  $x$  when these values are smaller than the current total time of the propagation (with a security margin).

## 5. Mesh adaptation for a sharper approximation of the signed distance function

We approximate the signed distance function to  $\Omega$  by means of a  $\mathbb{P}^1$  finite elements function ; therefore it seems natural to attempt to decrease the interpolation error of this function on the background mesh  $\mathcal{T}$ . Moreover, if we intend to use the signed distance function to  $\Omega$  as the initial implicit function defining  $\Omega$  when using level set methods, we may want the 0 level set of the approximated signed distance function (which is intended to be close to the  $\mathbb{P}^1$  interpolant of this function) to match as much as possible the true boundary  $\partial\Omega$ . Thus it can be interesting to couple the computation of the signed distance function with a process of adaptation of the background mesh  $\mathcal{T}$ . Actually, we will see that adapting  $\mathcal{T}$  the same way leads to an improvement as regards both problems.

### 5.1. Anisotropic mesh adaptation

Mesh adaptation basically lies on the fact that the most efficient way to refine a mesh so as to increase the efficiency of computations is to adjust the directions and sizes of its

elements in agreement with the variations of the functions under consideration. Thus, significant improvements can be achieved in accuracy, while the cost of the computations (related to the total number of elements of the mesh) is kept minimal, and conversely.

Numerous methods exist as regards the very popular topic of mesh adaptation, some of them relying on the concepts of Riemannian metric [21]: the local desired size, shape and orientation related information at a node  $x$  of mesh  $\mathcal{T}$  is stored in a metric tensor field  $M(x)$ , prescribed by an error indicator or an error estimate which can arise from various possible preoccupations : a posteriori geometric error estimates, analytic error estimates, etc... (see for instance [2], [5], [22]).

Given a metric tensor field  $M(x)$  defined at each point  $x \in \mathbb{R}^d$ , (notice that in practice,  $M(x)$  is defined only at the nodes of  $\mathcal{T}$  and then interpolated from these values [16]) we consider respectively the *length*  $l_M(\gamma)$  of a curve  $\gamma : [0, 1] \rightarrow \mathbb{R}^d$  and the *volume*  $V_M(K)$  of a simplex  $K$  in the Riemannian space  $(\mathbb{R}^d, M)$ :

$$l_M(\gamma) = \int_0^1 \sqrt{\langle M(\gamma(t))\gamma'(t), \gamma'(t) \rangle} dt, \quad V_M(K) = \int_K \sqrt{\det(M(x))} dx$$

and aim at modifying the mesh  $\mathcal{T}$  so as to make it *quasi-unit* with respect to the metric  $M(x)$ , that is to say all its simplices  $K$  have edges lengths lying in  $\left[\frac{1}{\sqrt{2}}, \sqrt{2}\right]$  and an *anisotropic quality measure*:

$$\mathcal{Q}_M(K) := \alpha_d \frac{V_M(K)^2}{\left(\sum_{i=1}^{na} l_M(e_i)^2\right)^d}$$

close to 1 (where  $na = d(d+1)/2$  is the number of edges of a  $d$ -dimensional simplex,  $e_i$  are the edges of  $K$  and  $\alpha_d$  is a normalization factor). For instance, in the particular case when  $M(x)$  is constant over  $\mathbb{R}^d$ ,  $M$  can be characterized by the ellipsoid

$$\Phi_M(1) = \{v \in \mathbb{R}^d / l_M(v) = 1\}$$

of unit vectors with respect to  $M$ , and a simplex  $K$  with unit edges is simply a simplex enclosed in this ellipsoid.

Several techniques have been devised for generating anisotropic meshes according to a metric tensor field, that can be roughly classified into two categories. On the one hand, global methods, such as Delaunay-based methods and advancing-front methods, perform the same kind of operations as in the classical case with adapted notions of length and volume. On the other hand, local mesh modification methods [17] start from an existing non-adapted mesh and adapt it so that it fits at best the above conditions. The approach used in this paper belongs to the second category.

### 5.2. Computation of a metric tensor associated to the minimization of the $\mathbb{P}^1$ interpolation error

Let us recall the following result (whose proof lies in [13] [4]) about the interpolation error of a smooth function  $u$  on a simplicial mesh by means of a  $\mathbb{P}^1$  finite elements function, in  $L^\infty$  norm.

**Theorem 5.1.** *Let  $\mathcal{T}$  a simplicial mesh of a polyhedral domain  $\mathcal{D} \subset \mathbb{R}^d$  ( $d = 2$  or  $3$ ) and  $u$  a  $\mathcal{C}^2$  function on  $\mathcal{D}$ . Let  $V_{\mathcal{T}}$  the space of continuous functions on  $\mathcal{D}$  whose restriction to every simplex of  $\mathcal{T}$  is a  $\mathbb{P}^1$  function, and denote by  $\pi_{\mathcal{T}} : \mathcal{C}(\mathcal{D}) \rightarrow V_{\mathcal{T}}$  the usual  $\mathbb{P}^1$  finite elements interpolation operator. Then for every simplex  $K \in \mathcal{T}$ ,*

$$\|u - \pi_{\mathcal{T}}u\|_{L^\infty(K)} \leq \frac{1}{2} \left( \frac{d}{d+1} \right)^2 \max_{x \in K} \max_{y, z \in K} \langle |\mathcal{H}(u)|(x)yz, yz \rangle$$

where, for a symmetric matrix  $S \in \mathcal{S}_d(\mathbb{R})$ , which admits the following diagonal shape in

orthonormal basis  $S = P \begin{pmatrix} \lambda_1 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \lambda_d \end{pmatrix} {}^tP$ , we denote

$$|S| := P \begin{pmatrix} |\lambda_1| & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & |\lambda_d| \end{pmatrix} {}^tP.$$

Relying on this theorem, we build a metric tensor field  $M(x)$  on  $\mathcal{D}$  that allows an accurate control of the  $L^\infty$  interpolation error of  $u$  (recall the main feature of this interpolation error is that it is an upper bound for the approximation error of  $u$  by means of a finite elements calculus with the space  $V_{\mathcal{T}}$ , at least in the case of elliptic problems [10]): so as to get

$$\forall K \in \mathcal{T}, \|u - \pi_{\mathcal{T}}u\|_{L^\infty(K)} \leq \epsilon$$

for a prescribed margin  $\epsilon > 0$ . We urge the shape of an element  $K$  of  $\mathcal{T}$  to behave in such a way that

$$\max_{y, z \in K} \langle \tilde{\mathcal{H}}yz, yz \rangle \leq c\epsilon$$

where  $c$  is a constant which only depends on the dimension, stemming from theorem (5.1) and  $\tilde{\mathcal{H}}$  is the mean value (or an approximation) of the metric tensor  $|\mathcal{H}(u)|$  over element  $K$ . This leads to defining the desired metric tensor  $M(x)$  at each node  $x$  of  $\mathcal{T}$  by (see [1] [2]):

$$M(x) = P(x) \begin{pmatrix} \tilde{\lambda}_1 & 0 & 0 \\ 0 & \tilde{\lambda}_2 & 0 \\ 0 & 0 & \tilde{\lambda}_3 \end{pmatrix} {}^tP(x) \quad (13)$$



where

$$\tilde{\lambda}_i = \min(\max(\frac{c|\lambda_i|}{\epsilon}, \frac{1}{h_{max}^2}), \frac{1}{h_{min}^2}), \quad \widetilde{|\mathcal{H}(u)|}(x) = P(x) \begin{pmatrix} |\lambda_1| & 0 & 0 \\ 0 & |\lambda_2| & 0 \\ 0 & 0 & |\lambda_3| \end{pmatrix} {}^t P(x)$$

being an approximation of the Hessian of  $u$  around node  $x$ , written here in diagonal form in an orthonormal basis,  $h_{min}$  (resp.  $h_{max}$ ) being the smallest (resp. largest) size allowed for an element in any direction, and  $c$  being the above constant.

### 5.3. Mesh adaptation for a geometric reconstruction of the 0 level set of a function

In this section, we consider a bounded domain  $\Omega \subset \mathbb{R}^d$ , implicitly defined by a function  $u$  - that is,  $\Omega = \{x \in \mathbb{R}^d / u(x) < 0\}$  and  $\partial\Omega = \{x \in \mathbb{R}^d / u(x) = 0\}$  - and we want to adapt the background mesh  $\mathcal{T}$  so that the 0-level set of the function  $\pi_{\mathcal{T}}u_{\Omega}$  obtained from  $u$  by  $\mathbb{P}^1$  finite elements interpolation, say  $\partial\Omega_{\mathcal{T}}$ , is as close as possible to the 0-level set of the true function  $u$ , in terms of the Hausdorff distance (for a review of the various methods for discretizing an implicit surface, see [19]). Then we will apply these results to the particular case when  $u$  is the signed distance function to  $\Omega$ . Throughout this section, we follow the previous work in [14].

**Definition 5.1.** Let  $K_1, K_2$  two compact subsets of  $\mathbb{R}^d$ . For all  $x \in \mathbb{R}^d$ , denote  $d(x, K_1) = \inf_{y \in K_1} d(x, y)$  the Euclidean distance from  $x$  to  $K_1$ . We define :

$$\rho(K_1, K_2) := \sup_{x \in K_1} d(x, K_2)$$

and call Hausdorff distance between  $K_1$  and  $K_2$ , denoted by  $d^H(K_1, K_2)$ , the nonnegative real number

$$d^H(K_1, K_2) := \max(\rho(K_1, K_2), \rho(K_2, K_1)).$$

The following small lemma will come in handy when we have to measure the distance to  $\Omega$  thanks to  $u$  :

**Lemma 5.2.** Let  $u$  a  $\mathcal{C}^1$  function on  $\mathbb{R}^d$ , without any critical point, so that the set  $\partial\Omega$  is a submanifold of  $\mathbb{R}^d$ , and  $\Omega$  is a bounded subdomain of  $\mathbb{R}^d$  with  $\mathcal{C}^1$  boundary. For any point  $x \in \mathbb{R}^d$  we have the estimate :

$$d(x, \partial\Omega) \leq \frac{\sup_{z \in \mathbb{R}^d} \|\nabla u(z)\|}{\inf_{z \in \mathbb{R}^d} \|\nabla u(z)\|^2} |u(x)| \quad (14)$$

*Proof.* The proof consists in 'going backwards' following the velocity field  $\nabla u$  until going against  $\partial\Omega$ , then to estimate the distance between  $x$  and the contact point with  $\partial\Omega$ . To

achieve this, introduce the characteristic curve  $\gamma$  associated to the field  $\nabla u$  and starting from  $x$ , defined as a solution of the EDO :

$$\begin{cases} \gamma(0) = x \\ \gamma'(s) = \nabla u(\gamma(s)) \end{cases} \quad (15)$$

Owing to arguments similar to those used in section 3, the curve  $s \mapsto \gamma(s)$  is defined for all real values  $s$  - including negative ones - and there exists a real number  $s_0$ , positive or negative if, respectively  $x \in \Omega$  or  $x \in {}^c\Omega$ , such that  $y := \gamma(s_0)$  belongs to  $\partial\Omega$ . We then have :

$$\begin{aligned} u(x) &= u(x) - u(y) \\ &= \int_{s_0}^0 \langle \nabla u(\gamma(s)), \gamma'(s) \rangle ds \\ &= \int_{s_0}^0 ||\nabla u(\gamma(s))||^2 ds \end{aligned}$$

and thus obtain

$$|s_0| \inf_{z \in \mathbb{R}^d} ||\nabla u(z)||^2 \leq |u(x)| \quad (16)$$

On the other hand, we also get

$$x - y = \int_{s_0}^0 \nabla u(\gamma(s)) ds$$

and

$$||x - y|| \leq |s_0| \sup_{z \in \mathbb{R}^d} ||\nabla u(z)||. \quad (17)$$

Note that, in practice, the computational domain  $\mathbb{R}^d$  is restricted to a bounding box containing  $\Omega$ , so that taking supremum or infimum in this framework does not pose any problem. Eventually, with (16) and (17), we conclude that :

$$d(x, \partial\Omega) \leq ||x - y|| \leq \frac{\sup_{z \in \mathbb{R}^d} ||\nabla u(z)||}{\inf_{z \in \mathbb{R}^d} ||\nabla u(z)||^2} |u(x)|.$$

□

Note that formula (15) expresses the idea that the closer  $u$  is to the signed distance function to  $\Omega$ , the more we can rely on the evaluation of  $u$  to estimate the Euclidean distance to the boundary  $\partial\Omega$ .

We are now ready to estimate the Hausdorff distance between  $\partial\Omega$  and  $\partial\Omega_{\mathcal{T}}$ . Take a point  $x \in \partial\Omega_{\mathcal{T}}$ , which belongs to a simplex  $K$  of the background mesh  $\mathcal{T}$ . With lemma 5.2 we

have

$$\begin{aligned} d(x, \partial\Omega) &\leq \frac{\sup_{y \in \mathbb{R}^d} \|\nabla u(y)\|}{\inf_{y \in \mathbb{R}^d} \|\nabla u(y)\|^2} |u(x)| \\ &= \frac{\sup_{y \in \mathbb{R}^d} \|\nabla u(y)\|}{\inf_{y \in \mathbb{R}^d} \|\nabla u(y)\|^2} |u(x) - \pi_{\mathcal{T}} u(x)| \end{aligned}$$

and it yields, thanks to theorem 5.1

$$d(x, \partial\Omega) \leq c \frac{\sup_{y \in \mathbb{R}^d} \|\nabla u(y)\|}{\inf_{y \in \mathbb{R}^d} \|\nabla u(y)\|^2} \max_{x \in K} \max_{y, z \in K} \langle |\mathcal{H}(u)|(x) yz, yz \rangle$$

where  $c$  is a scalar constant which only depends on  $d$ . Thus, we find

$$\rho(\partial\Omega_{\mathcal{T}}, \partial\Omega) \leq c \frac{\sup_{y \in \mathbb{R}^d} \|\nabla u(y)\|}{\inf_{y \in \mathbb{R}^d} \|\nabla u(y)\|^2} \max_{K \in \mathcal{T}} \max_{x \in K} \max_{y, z \in K} \langle |\mathcal{H}(u)|(x) yz, yz \rangle.$$

By the same token, applied to a point  $x \in \partial\Omega$ , we have

$$\rho(\partial\Omega, \partial\Omega_{\mathcal{T}}) \leq c \frac{\sup_{K \in \mathcal{T}} \|\nabla u_{\mathcal{T}}|_K\|}{\inf_{K \in \mathcal{T}} \|\nabla u_{\mathcal{T}}|_K\|^2} \max_{K \in \mathcal{T}} \max_{x \in K} \max_{y, z \in K} \langle |\mathcal{H}(u)|(x) yz, yz \rangle.$$

Eventually, neglecting the discrepancy between  $\|\nabla u\|$  and  $\|\nabla u_{\mathcal{T}}\|$  yields :

$$d^H(\partial\Omega, \partial\Omega_{\mathcal{T}}) \leq c \frac{\sup_{y \in \mathbb{R}^d} \|\nabla u(y)\|}{\inf_{y \in \mathbb{R}^d} \|\nabla u(y)\|^2} \max_{K \in \mathcal{T}} \max_{x \in K} \max_{y, z \in K} \langle |\mathcal{H}(u)|(x) yz, yz \rangle.$$

We observe that this estimate is very similar to the result given by theorem 5.1, and especially that it involves the Hessian matrix of  $u$  ; thus, with the same metric tensor field (13) as the one associated to the control of the  $\mathbb{P}^1$ -interpolation error on  $\mathcal{T}$ , we achieve control of the Hausdorff distance between the exact boundary  $\partial\Omega$  and its piecewise affine reconstruction  $\partial\Omega_{\mathcal{T}}$ . Of course, depending on where we need the accuracy, we can restrict ourselves to prescribing metric (13) only in certain areas of  $\mathbb{R}^d$  (e.g. in a level set context, in a narrow band near the boundary, or in the vicinity of a particular isosurface of  $u$ ).

This result admits a rather interesting geometric interpretation in the case when  $u = u_{\Omega}$ , i.e. when  $u$  is a signed distance function. In that case, it is well-known [8] the second fundamental form reads, for any point  $x \in \partial\Omega$  :

$$\forall \xi \in T_x \partial\Omega, \quad \mathbf{II}_x(\xi, \xi) = -\langle \mathcal{H}u(x) \xi, \xi \rangle$$

where  $\langle ., . \rangle$  denotes the usual Euclidean scalar product of  $\mathbb{R}^d$ . Hence, up to the sign, the eigenvalues of  $\mathcal{H}u(x)$  are the two principal curvatures of the surface  $\partial\Omega$ , associated with the two principal directions at point  $x$ . In such case, the above estimates mean that, understandingly enough, so as to get the best reconstruction of  $\partial\Omega$ , we have to align the circumscribed ellipsoids of the simplices of the background mesh  $\mathcal{T}$  with the curvature of this surface (see figure (3)).

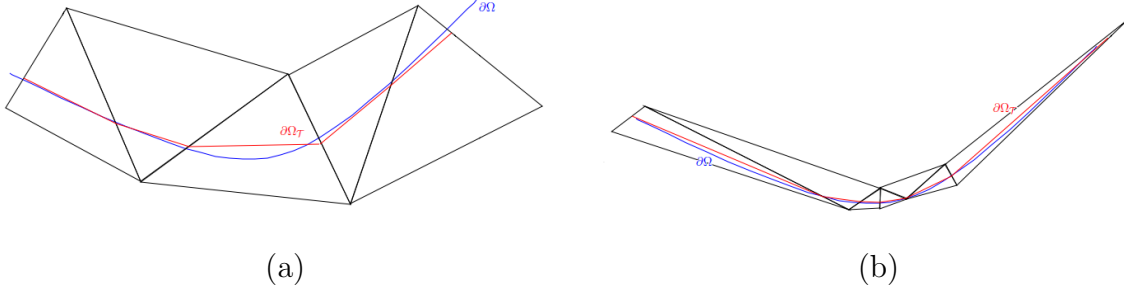


Figure 3:  $\mathbb{P}^1$ -reconstruction of  $\partial\Omega$  with (a) a regular background mesh  $\mathcal{T}$  and (b) an adapted anisotropic background mesh  $\mathcal{T}$ .

## 6. A remark about level-set redistanciation

When evolving an interface by means of the level set method, a crucial issue is to maintain the level set function  $\phi^n = \phi(t^n, .)$  ( $t^n = n\Delta t$ , where  $\Delta t$  is the time step of the process) as close as possible to the signed distance function to the zero level set  $\partial\Omega^n$  so defined (see [30]), while it tends to get very far from it in few iteration steps. To achieve this, a redistanciation step must be carried out [36]. Unfortunately, it is worth emphasizing this problem is ill-posed since it is impossible to "regularize" function  $\phi^n$  to make it close to a distance function  $\widetilde{\phi}^n$  - i.e.  $||\nabla \widetilde{\phi}^n|| \approx 1$  - without changing the zero level-set which then becomes  $\widetilde{\partial\Omega}^n := \{x \in \mathbb{R}^d / \widetilde{\phi}^n(x) = 0\}$ . Several approaches exist to address this issue, depending on the applications they are suited for and thus on the features of the interface  $\partial\Omega^n$  which must be retained (see e.g. [27] [39] [38] [30]).

Hence, we limit ourselves to the case of an adaptation process, where at each step, the background mesh  $\mathcal{T}$  is adapted to the zero level-set of function  $\phi^n$ , by prescribing the metric tensor field (13) at least in a narrow band near the interface (see [9] for an example). Here, implementing the redistanciation step  $\phi^n \rightarrow \widetilde{\phi}^n$  by taking for  $\widetilde{\phi}^n$  the signed distance function to  $\Omega^n$ , computed by means of the algorithm devised in section (4), provides an true signed distance function  $\widetilde{\phi}^n$ , while ensuring the movement of the interface is controlled by :

$$d^H(\partial\Omega^n, \widetilde{\partial\Omega}^n) \leq \epsilon.$$

## 7. Extension to the computation of the signed distance function in a Riemannian space

The proposed method admits a straightforward extension in the case we want to compute the signed distance function  $u_\Omega^M$  from a subdomain  $\Omega$  of  $\mathbb{R}^d$ ,  $\mathbb{R}^d$  being endowed with a Riemannian metric  $M$ , that is,

$$u_\Omega^M(x) = \begin{cases} -d_M(x, \partial\Omega) & \text{if } x \in \Omega \\ 0 & \text{if } x \in \partial\Omega \\ d_M(x, \partial\Omega) & \text{if } x \in {}^c\bar{\Omega} \end{cases},$$

where  $d_M(x, y) = \inf_{\gamma(0)=x; \gamma(1)=y} l_M(\gamma)$  is the *distance* from  $x$  to  $y$  in the space  $(\mathbb{R}^d, M)$  and  $d_M(x, \partial\Omega) = \inf_{y \in \partial\Omega} d_M(x, y)$  is the *unsigned distance function* from  $x$  to  $\partial\Omega$ . For another approach, based on the Fast Marching Method, see [32] [25].

Indeed, the function  $u_\Omega^M$  is a solution of the *Eikonal equation* in the space  $(\mathbb{R}^d, M)$  in the sense of viscosity [26] :

$$\sqrt{\langle M^{-1}(x) \nabla u(x), \nabla u(x) \rangle} = 1$$

Considering a continuous function  $u_0$  which implicitly defines  $\Omega$  in the sense that (2) holds, we have the corresponding unsteady equation :

$$\begin{cases} \frac{\partial u}{\partial t}(t, x) + \text{sgn}(u_0(x))(\sqrt{\langle M^{-1}(x) \nabla u(t, x), \nabla u(t, x) \rangle} - 1) = 0 \\ u(t = 0, x) = u_0(x) \end{cases}$$

and the same study as in section (3) yields the following approximation formulae for computing the solution  $u$  of this equation : for  $t > 0$ , a small time step  $dt$ , and  $x \in {}^c\bar{\Omega}$  :

$$u(t + dt, x) \approx u \left( t, x - dt \frac{M^{-1}(x) \nabla u(t, x)}{\sqrt{\langle M^{-1}(x) \nabla u(t, x), \nabla u(t, x) \rangle}} \right) + dt$$

and, by symmetry, for  $x \in \Omega$  :

$$u(t + dt, x) \approx u \left( t, x + dt \frac{M^{-1}(x) \nabla u(t, x)}{\sqrt{\langle M^{-1}(x) \nabla u(t, x), \nabla u(t, x) \rangle}} \right) - dt$$

which raises a numerical scheme similar to the scheme presented in section 4 (see Figure 7 for an example).

## 8. Numerical Examples

Now, we provide several exemples to assess the main theoretical issues presented in the previous sections. All the following computations are held on contours embedded in a bounding box which is a unit square in two dimensions, or a unit cube in three dimensions, and are scaled if need be.

### Example 1

Here we give a 2-dimensional example of the computation of the signed distance function to the contour represented in Figure 8, which is quite interesting since it presents several connected components, whose identification is not utterly trivial. We first hold a computation on a very fine regular triangular mesh  $\mathcal{T}$  of the bounding box, so as to test the approximation of the signed distance function in several norms, on a nontrivial example, and on the whole computational domain. Its characteristic size is  $h \approx 0.002$ . Results - identification of the connected components, initialization of the sign and computation of the signed distance function - are shown in figure 8, and several features of the computation (mesh size, CPU time,...) are presented in Table 8. In this example, we chose a time step  $dt = 0.001$  - in accordance with the mesh size - for the steps  $1 \leq n \leq 80$ , and then finished the computation with a time step  $dt = 0.004$ . The exact signed distance function to this contour, with which we compare our approximation, is computed by a brute force approach - i.e. by computing the minimum distance to a segment of the mesh of the contour at every node of the background mesh  $\mathcal{T}$ . All our computations are carried out on an OPTERON 6100, 2Ghz.

Number of Nodes	Number of Elements	$\ e\ _{L^\infty}$	$\ e\ _{L^1}$	$\ e\ _{L^2}$	CPU Time (Seconds)
251,001	500,000	0.001862	0.000487	0.000598	13.85

Table 1: Computation of the signed distance function to the contour 8 on a regular mesh

Now, we adapt the background mesh  $\mathcal{T}$  to the computation of the signed distance function to the same contour, relying on the principles enonciated in section 5.1, in the vicinity of the boundary  $\partial\Omega$ . Therefore, we are only interested in getting a close approximation of this signed distance function near the boundary. We use parameters  $\epsilon = 0.001$  and  $h_{min} = 0.001$ . This yields the results of figure 8, whose features (on one CPU) are reported in table 8.

Number of Iterations	Number of Nodes	Number of Elements	CPU Time (Seconds)
236	8,030	16,006	0.42

Table 2: Computation of the signed distance function to the contour 8 on an adapted mesh.

### Example 2

We now turn to the three-dimensional case. Some details about the mesh sizes and CPU times are to be found in table 3. We tested our algorithm with the same procedure as before as regards the choice of the time step, and with a parallel implementation on 10 CPU, with a shared memory architecture. Note that only the stage corresponding to the propagation of the distance throughout the domain has been parallelized, and that the initialization step - which could also be easily parallelized - actually takes most of the computation time

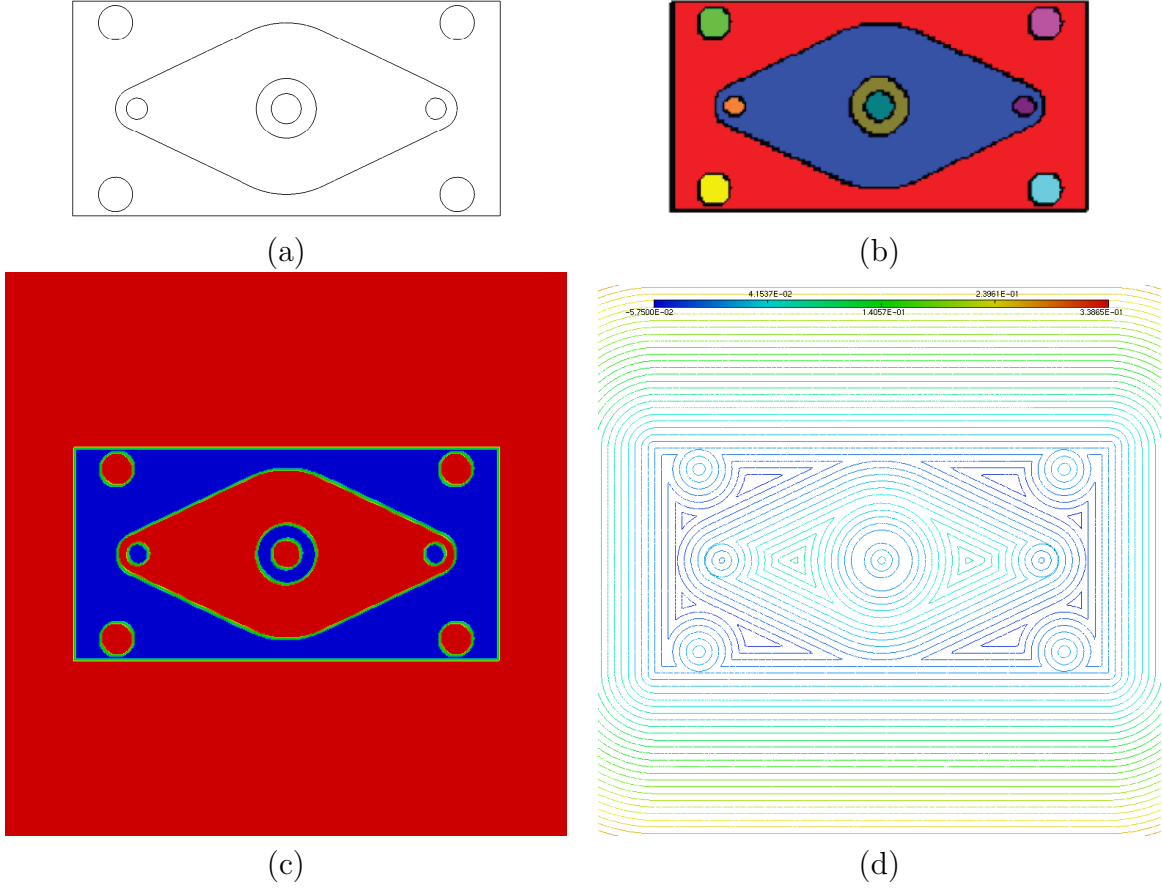


Figure 4: 2D example on a regular mesh ; (a) the 2D contour, (b) the connected components of the contour, (c) sign of these components : red for positive, blue for negative, green for the boundary, (d) some isolines of the signed distance function.

(Table 3 provides the number of faces of each initial contour so as to emphasize this feature). However, we thought it better to report in 3 the total computation time. First, we considered a mechanical part called *cimplex*. Figure 8 shows the original boundary, its reconstitution as the 0 level set of the computed approximation of the signed distance function to this contour, a level set of the computed function, and some cut in the adapted mesh. Note that the anisotropic feature of the background mesh  $\mathcal{T}$  allows a good approximation of the ridges of the contour, even though we did not apply any special process to achieve so. Actually, it is worth mentioning that some post-treatment could be implemented so as to recover *exactly* those sharp features in the reconstruction of the interface  $\partial\Omega$  [28], but we believe this goes beyond the scope of this paper.

The next example, hereafter named *wheel* (Figure 9), emphasizes another difficulty that may arise, especially when it comes to mechanical devices, exhibiting very fine structures (or more generally, very fine details). They are very difficult to track accurately when the background mesh  $\mathcal{T}$  is regular, the reason being the suitable size of a regular mesh for this purpose would be tremendous. Independantly, note that the surface mesh of the 0 level

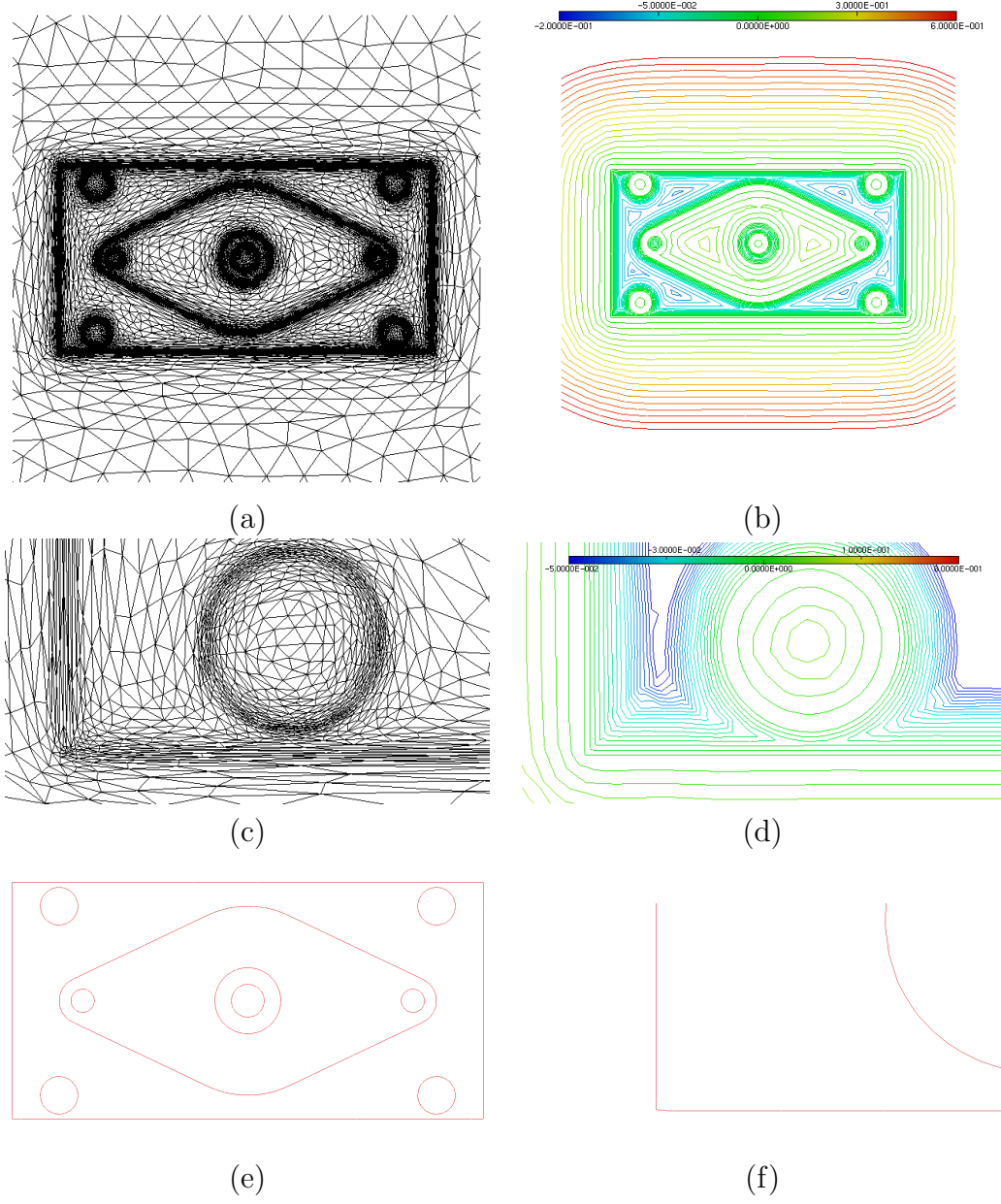


Figure 5: 2D example on an adapted mesh ; (a) the associated adapted mesh (b) some isolines of the signed distance function on this mesh (c) a detail in the background mesh (d) the corresponding detail among the isolines (e) the piecewise affine reconstructed contour and (f) a zoom on a corner of the reconstructed contour.

set of the computed signed distance function reconstructed by means of intersections with the background mesh  $\mathcal{T}$  may be very irregular and contain too much surface elements to allow any further calculation on it. To this end, it is often necessary to proceed to a surface remeshing step in order to generate a suitable computational mesh (see [16]). An example is provided in figure 9.

Then, Figure 8 gives another example, this of the computation of the signed distance



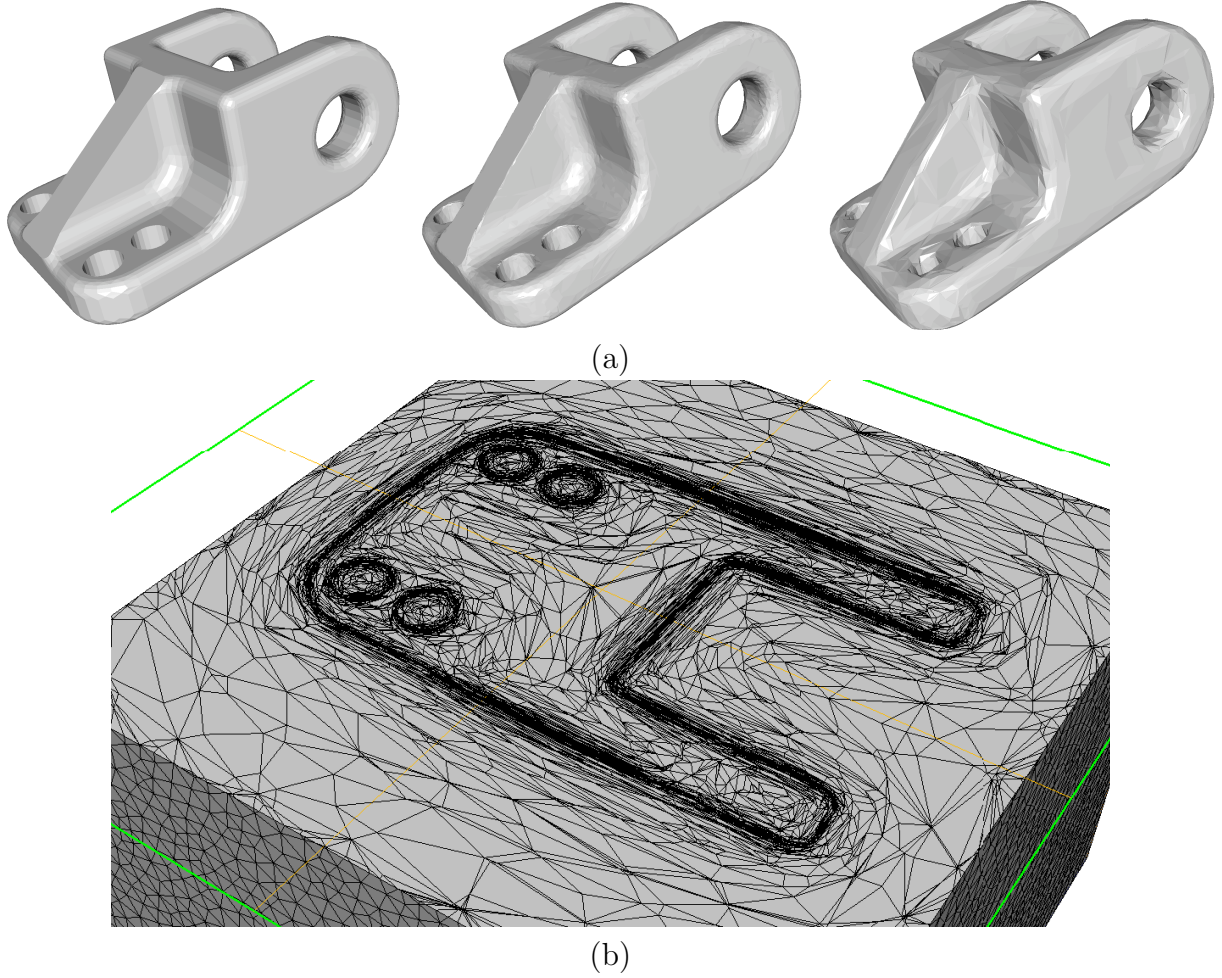


Figure 6: *Computation of the signed distance function to the 'cimplex' ; (a) Left -the initial domain, center - the reconstructed domain, right - isovalue 0.01 of the computed signed distance function, (b) A cut in the adapted mesh.*

function to the Stanford *Happy Buddha*, and of the approximation of this contour as the 0 level set of the computed function.

Eventually, in several cases, for very detailed contours, we thought necessary to perform an isotropic adaptation of the background mesh, and to make an intermediate computation of the signed distance function on it before indulging in an anisotropic adaptation of the background mesh, so as to make sure to capture any close detail of the contour. See Figure 10 and Table 3 for a comparison between both meshes.

### *Exemple 3*

So as to illustrate the concept introduced Section 7, our last example concerns the computation of the signed distance function to a domain  $\Omega \subset \mathbb{R}^2$ ,  $\mathbb{R}^2$  being endowed with the hyperbolic metric of the Poincaré half-plane : let  $\Omega$  be a union a 3 disks, embedded in the

	Number of Faces	Number of Iterations	Number of Nodes	Number of Elements	CPU Time (Seconds)
Cimplex	4,160	227	39,593	208,676	4.13
Happy Buddha	1,085,477	160	549,818	3,218,519	1,294
Wheel	19,972	194	367,236	2,065,767	46.641
Venus (isotropic)	63,940	212	2,265,359	13,482,983	225
Venus (anisotropic)	63,940	238	120,404	670,746	605

Table 3: Computation of the signed distance function to several subdomains of  $\mathbb{R}^3$ .

half-plane  $\mathbb{H} := \{(x, y) \in \mathbb{R}^2 | y > 0\}$  endowed with the so-called *Lobatchevski metric* defined by

$$\forall (x, y) \in \mathbb{H}, \quad M(x, y) := \frac{1}{y^2} I_2$$

where  $I_2$  stands for the unitary matrix of dimension 2. Figure 7 then show some level sets of the signed distance function to  $\Omega$  with respect to metric  $M$ .

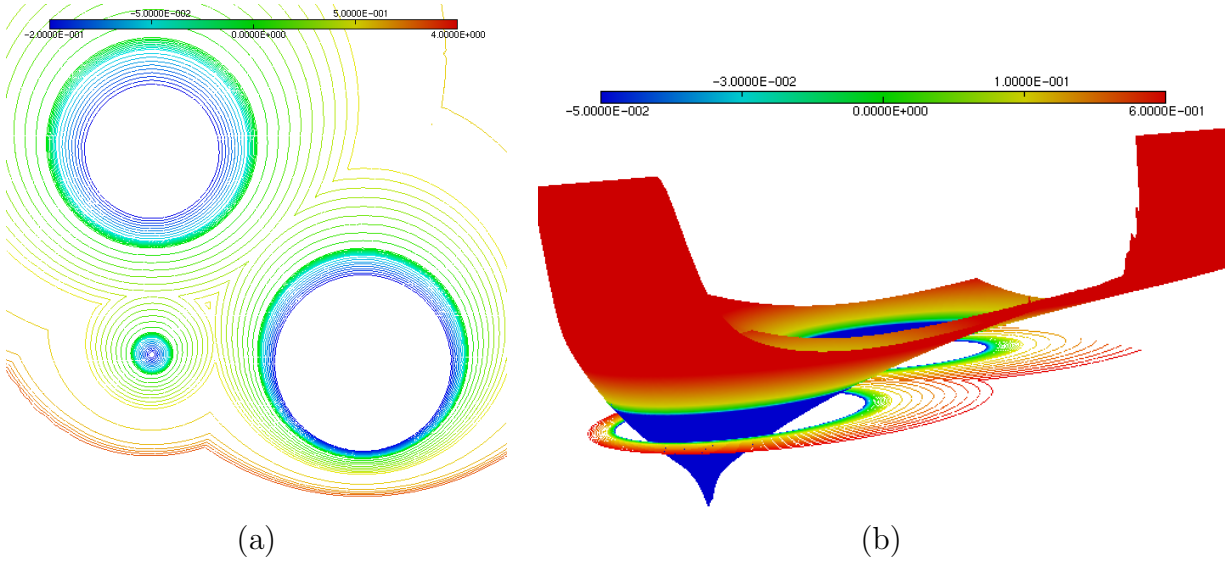


Figure 7: (a) Level sets of the signed distance function to three disks embedded in the Poincaré half-plane. (b) 3D graph of the function.

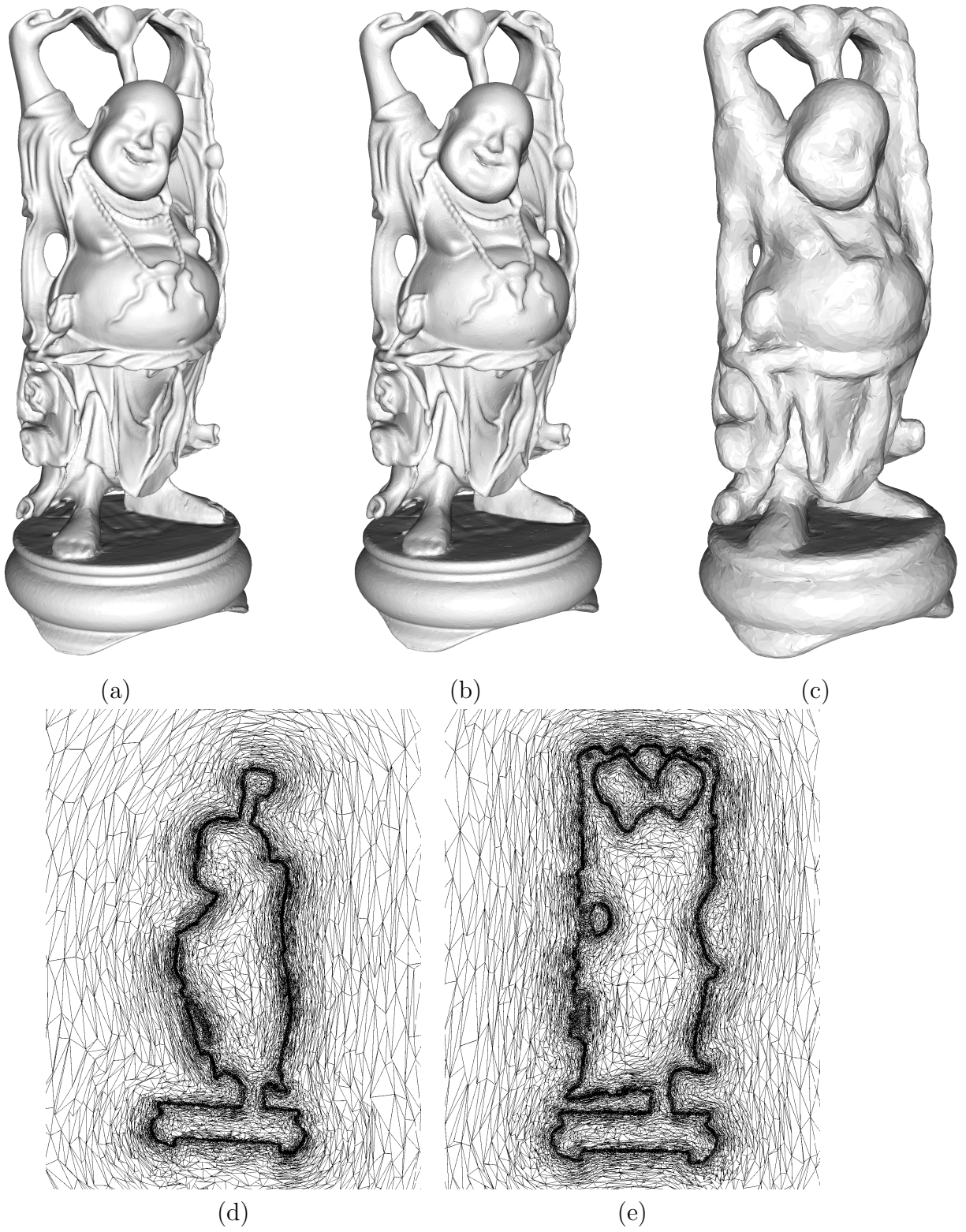


Figure 8: *Computation of the signed distance function to the Stanford "Happy Buddha" ; (a) The initial Buddha (b) isovalue 0.001 of the computed signed distance function (c) isovalue 0.01 (d) -(e) Two cuts in the adapted mesh.*

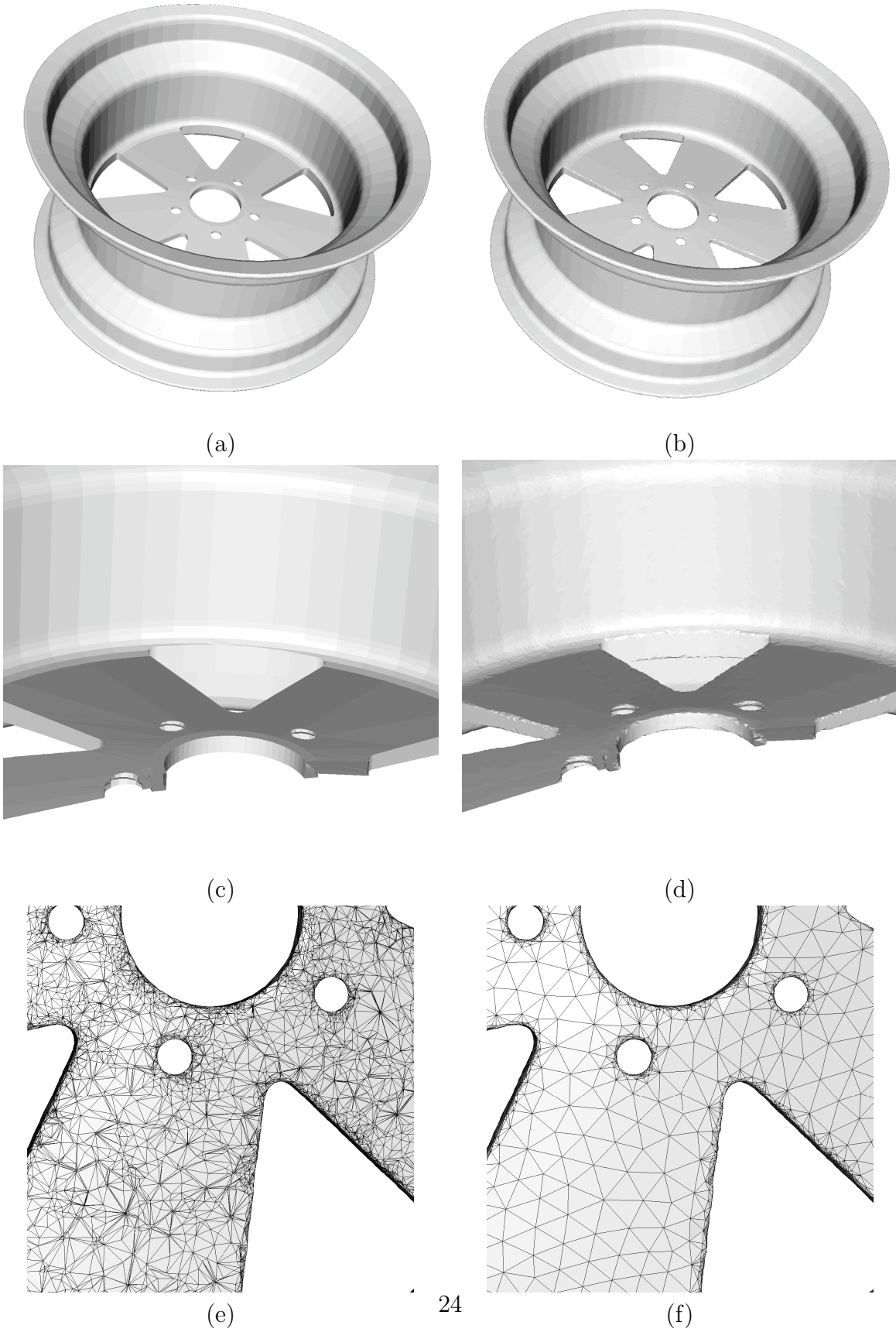


Figure 9: *Piecewise affine reconstruction of the wheel. (a) The original wheel (b) its reconstruction (c) a cut in the original wheel (d) a cut in the reconstruction (e) surface triangulation before remeshing (f) and after remeshing.*

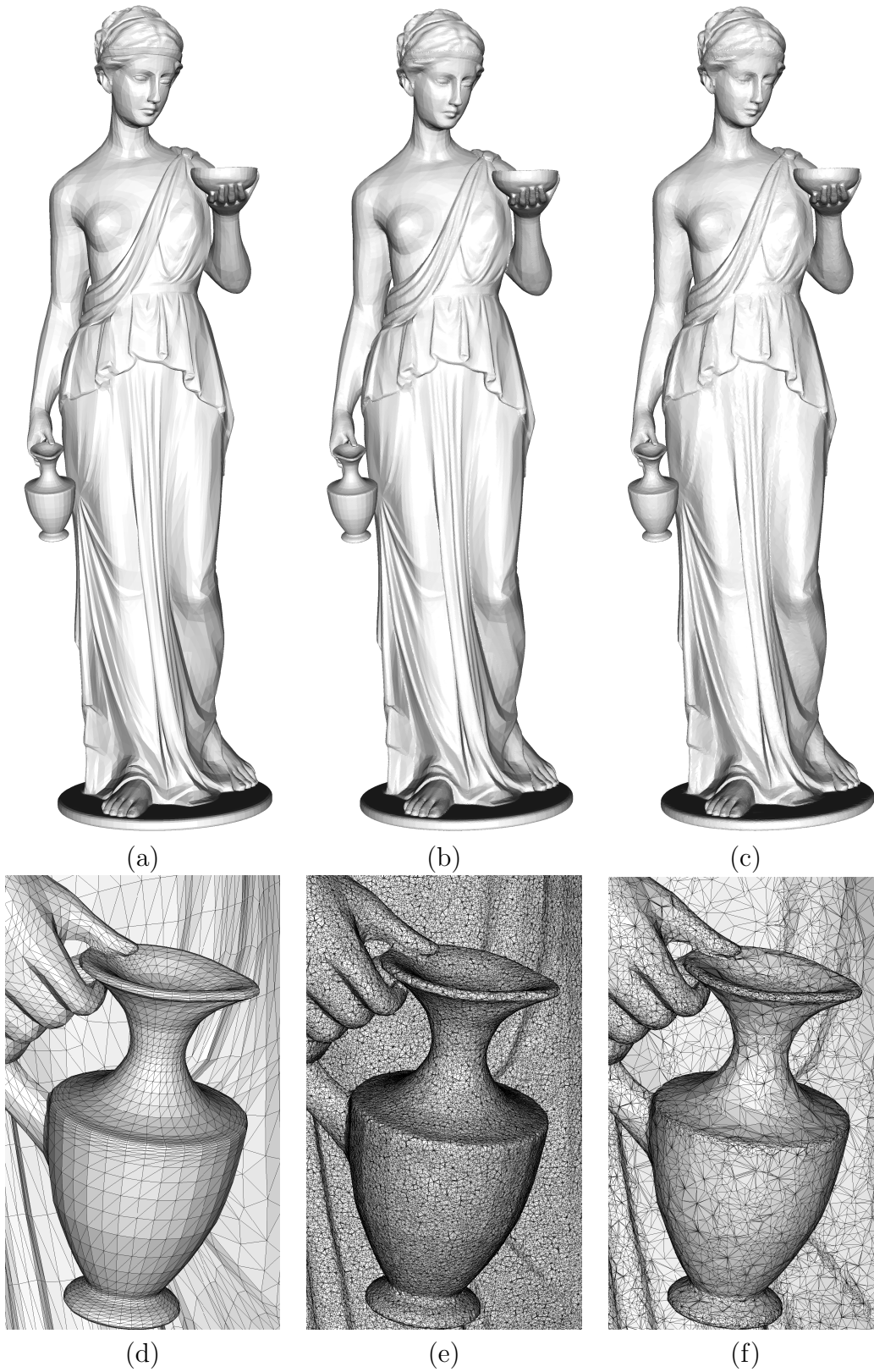


Figure 10: *Reconstruction of the "Venus" : (a) The original Venus (b) Its reconstruction with isotropic mesh adaptation (3093941 surface triangles) (c) Its reconstruction with anisotropic mesh adaptation (300968 surface triangles) (d) zoom on the original Venus (e) zoom on its reconstruction with isotropic mesh adaptation (f) zoom on its reconstruction with anisotropic mesh adaptation.*



## References

- [1] F. ALAUZET AND P. FREY, *Anisotropic Mesh Adaptation for CFD Computations*, Comput. Methods Appl. Mech. Engrg., 194 (2005), pp. 5068–5082.
- [2] F. ALAUZET AND P. FREY, *Estimateur d'erreur géométrique et métriques anisotropes pour l'adaptation de maillage. Partie I : aspects théoriques*, INRIA : Technical Report, 4759 (2003).
- [3] L. AMBROSIO, *Lecture Notes on geometric evolution problems, distance function and viscosity solutions.*, In Calculus of Variations and Partial Differential Equations, G.Buttazzo, A.Marino, M.K.V.Murthy Eds, Springer (1999), pp. 5-93.
- [4] M. V. ANGLADA, N.P. GARCIA AND P. B. CROSA, *Directional Adaptive Surface Triangulation*, Computer Aided Geometric Design, 16 (1999), pp. 107–126.
- [5] T. APEL, *Anisotropic Finite Elements : Local Estimates and Applications*, B.G. Teubner Stuttgart,Leipzig, Series of Advances in Numerical Mathematics, (1999).
- [6] V. ARNOLD, *Mathematical Methods of Classical Mechanics*, Springer, (1974).
- [7] J.F. AUJOL AND G. AUBER, *Signed distance functions and viscosity solutions of discontinuous Hamilton-Jacobi Equations*, INRIA Technical Report, 4507 (2002).
- [8] M. BERGER AND B. GOSTIAUX, *Differential Geometry : Manifolds, Curves and Surfaces*, Graduate Texts in Mathematics, Springer-Verlag (1987).
- [9] T.T.C. BUI, P. FREY AND B. MAURY, *A coupling strategy for solving two-fluid flows*, Int. J. Numer. Methods in Fluids, to appear (2010).
- [10] P.G. CIARLET, *The Finite Element Method for Elliptic Problems*, North Holland Publishing Company, (1978).
- [11] A. CLAISSE AND P. FREY, *Level Set Driven Smooth Curve Approximation From Unorganized or Noisy Point Set*, ESAIM : Proceedings,(2008).
- [12] M.G. CRANDALL, H.ISHII AND P.L. LIONS, *User's guide to viscosity solutions of second order partial differential equations*, Bulletin of the American Mathematical Society, 27 (1992), pp. 1–67.
- [13] E.F. D'AZEVEDO AND R.B. SIMPSON, *On Optimal Triangular Meshes for Minimizing the Gradient Error*, Numerische Mathematik, 59 (1991), pp. 321–348.
- [14] V. DUCROT AND P. FREY, *Contrôle de l'approximation géométrique d'une interface par une métrique anisotrope*, Comptes Rendus de l'Académie des Sciences, 345 (2007), pp. 537–542.
- [15] L.C. EVANS, *Partial Differential Equations*, American mathematical society,(1997).

- [16] P.J. FREY AND P.L. GEORGE, *Mesh Generation : Application to Finite Elements*, Wiley, 2nd Edition, (2008).
- [17] C. DOBRZYNSKI AND P. FREY, *Anisotropic Delaunay Mesh Adaptation for Unsteady Simulations*, Proc. of the 17th IMR, (2008), pp. 177–194.
- [18] A. FUHRMANN, G. SOBOTTKA AND C. GROSS, *Abstract Distance Fields for Rapid Collision Detection in Physically Based Modeling*, International Conference Graphicon : Proceedings,(2003).
- [19] A.J.P GOMES, I. VOICULESCU, J. JORGE, B. WYVILL AND C. GALLSBRAITH, *Implicit Curves and Surfaces : Mathematics, Data Structures and Algorithms*, Springer,(2009).
- [20] P. GUIGUE AND O. DEVILLERS, *Fast and Robust Triangle-Triangle Overlap Test using Orientation Predicates*, Journal of Graphics, Gpu, and Game Tools, 8 (2003), pp. 25–42.
- [21] M.G. VALLET, F. HECHT AND B. MANTEL, *Anisotropic Control of Mesh Generation Based upon a Voronoi Type Method*, Numerical Grid Generation in Computational Fluid Dynamics and Related Fields, (1991).
- [22] W. HUANG, *Metric Tensors for Anisotropic Mesh Generation*, Journal of Computational Physics, 204 (2005), pp. 633–665.
- [23] H. ISHII, *Existence and uniqueness of solutions of Hamilton-Jacobi equations*, Funkcialaj Ekvacioj, 29 (1986), pp. 167–188.
- [24] M.W. JONES, *3D distance from a Point to a Triangle*, Department of Computer Sciences, University of Wales Swansea : technical report, (1995).
- [25] R. KIMMEL AND J.A. SETHIAN, *Computing Geodesic Paths on Manifolds*, Proc. Nat. Acad. Sci. , 95 (1998), pp. 8431–8435.
- [26] C. MANTEGAZZA AND A.C. MENNUCCI, *Hamilton-Jacobi Equations and Distance Functions on Riemannian Manifolds*, Appl. Math. Opt., 47 (2003), pp. 1–25.
- [27] F. MUT, G.C. BUSCAGLIA AND E. A. DARI, *A New Mass-Conserving Algorithm for Level-Set Redistancing on Unstructured Meshes* , Mecanica Computational, 23 (2004), pp. 1659–1678.
- [28] D.L. PAGE, Y. SUN, A.F. KOSCHAN, J. PAIK AND M.A. ABIDI, *Normal Vector Voting: Crease Detection and Curvature Estimation on Large, Noisy Meshes* , Graphical Models, 64 (2004), pp. 199–229.
- [29] J. QIAN, Y.T. ZHANG AND H. ZHAO, *Fast Sweeping Methods for Eikonal Equations on Triangular Meshes*, SIAM J. Numer. Anal., 45 (2007), pp. 83–107.

- [30] S. J. OSHER AND R. FEDKIW, *Level Set Methods and Dynamic Implicit Surfaces*, Springer, (2002).
- [31] S.J. OSHER AND J.A. SETHIAN, *Fronts Propagating with Curvature-Dependent Speed: Algorithms Based on Hamilton–Jacobi Formulations*, Journal of Computational Physics, 79 (1988), pp. 12–49.
- [32] F. MEMOLI AND G. SAPIRO, *Fast Computation of Weighted Distance Functions and Geodesics on Implicit Hyper-Surfaces*, Journal of Computational Physics, 173 (2001), pp. 730–764.
- [33] J.A. SETHIAN, *Fast Marching Methods*, SIAM Review, 41 (1999), pp. 199–235.
- [34] J.A. SETHIAN, *A Fast Marching Method for Monotonically Advancing Fronts*, Proc. Nat. Acad. Sci., 93 (1996), pp. 1591–1595.
- [35] J.A. SETHIAN, *Level Set Methods and Fast Marching Methods : Evolving Interfaces in Computational Geometry, Fluid Mechanics, Computer Vision, and Materials Science*, Cambridge University Press, (1999).
- [36] S. J. OSHER, P. SMEREKA AND M. SUSSMAN, *A Level Set Approach for Computing Solutions to Incompressible Two-phase Flow*, Journal of Computational Physics, 114 (1994), pp. 146–159.
- [37] J. STRAIN, *Fast Tree-Based Redistancing for Level Set Computations*, Journal of Computational Physics, 152 (1999), pp. 664–686.
- [38] M. SUSSMAN AND E. FATEMI, *An Efficient, Interface-Preserving Level Set Redistancing algorithm and its Applications to Interfacial Incompressible Fluid Flow*, SIAM J. Sc. Comp., 20 (1999), pp. 1165–1191.
- [39] M. SUSSMAN, E. FATEMI AND P. SMEREKA AND S. OSHER, *An Improved Level-Set Method for Incompressible Two-Phase Flows*, Computers and Fluids, 27 (1997), pp. 663–680.
- [40] H. ZHAO, *A Fast Sweeping Method for Eikonal Equations*, Math. Comp., 74 (2005), pp. 603–627.
- [41] H. ZHAO, S.J. OSHER AND R. FEDKIW, *Fast Surface Reconstruction Using the Level Set Method*, Proceedings of IEEE Workshop on Variational and Level Set Methods in Computer Vision, (2001).